Stage 2

This stage of the project you will provide a data component which will be used by the **Control** components of the MVC framework to gather data required by the JavaServer Pages and populate that data into Java Bean data stores. Then the JavaServer Pages will retrieve the information from the Java Bean data stores and display it on the web page. By this point we will not have covered the database access technology, **JDBC**, so the data component will use *stub programming* to fake the data. Your **Control** should use the classes that implement the **MaterialDAO**, **MaterialReservationDAO**, and **AccountOwnerDAO** interfaces to generate data where possible. Note that since the methods of these classes have a **throws SQLException** clause, you will need to use a **try...catch** to catch the SQLException and then wrap it in a ServletException and throw it (e.g. `throw new ServletException(sqle);` where sqle is the SQLException object). Use the **MaterialDAOFactory**, **MaterialReservationDAOFactory**, and **AccountOwnerDAOFactory** classes to generate instances of the data access objects.

Important: Do not share your or your partner's java code with anyone.

This stage is worth **5%** of the 15% project mark. Submit both parts (Group A and Group B) in a single Eclipse Dynamic Web Project. Zip the entire project folder. Submit the project to the submission drop box by 11:59PM on ~~Tuesday November 17, 2015~~ **Thursday November 19, 2015**. Be sure to include the your name *within* the XHTML files that you author (as XHTML <-- comment --> tags) and as java comments (// comment) within your java source files.

The Java Bean data store classes have been developed for you and are available in the file limarabeans.jar. See the api for a description of the classes.

The Data Access Object interfaces have been developed for you and are available in the file limaradao.jar. See the api for a description of the interfaces.

You may use any of the classes available in the ca.on.senecac.prg556.common package. The classes are available in the file common.jar. See the api for a description of the classes.

Group A Components

Members of this group are responsible for creating the classes **MaterialDAOFactory**, **MaterialData**, **AccountOwnerDAOFactory**, and **AccountOwnerData** and stubbing the methods to provide data. The MVC **Control** classes for this group are also specified. These classes will need to use the **MaterialDAOFactory**, **MaterialReservationDAOFactory**, and **AccountOwnerDAOFactory** classes to create the **MaterialDAO**, **MaterialReservationDAO**, and **AccountOwnerDAO** objects as needed. The classes are described below.

The `MaterialDAOFactory` Class

Create this class in a **data** sub-package under your project package.

Methods

`public static MaterialDAO getMaterialDAO()`
Returns a MaterialDAO object.

The `MaterialData` Class

The `MaterialData` class must implement the methods of the MaterialDAO interface. Create this class in a **data** sub-package under your project package. **This class should not be public** (use package level access).

Methods

`public List<Material> getAvailableMaterials(String title, String type) throws SQLException` (see interface description)
For stage 2, this method should create a list containing two of the three library material objects used in the **MaterialData.getMaterial** method below (hint: call the getMaterial method to create the Material objects).

`public Material getMaterial(String materialId) throws SQLException` (see interface description)
For stage 2, this method should create one of three possible library material objects based on the material id provided. The method will compare the material id parameter against three material ids hardcoded into the method. If the specified material id does not match up with any of the hard coded values, then the method should return **null**. Otherwise, the method should create and return a library material object using the given material id and hardcoded title and type values. The library material type should be a value like **Book**, **Audio**, **Video**, etc.

The `AccountOwnerDAOFactory` Class

Create this class in a **data** sub-package under your project package.

Methods

`public static AccountOwnerDAO getAccountOwnerDAO()`
Returns a AccountOwnerDAO object.

The `AccountOwnerData` Class

The `AccountOwnerData` class must implement the methods of the AccountOwnerDAO interface. Create this class in a **data** sub-package under your project package. **This class should not be public** (use package level access).

Methods

`public AccountOwner createLibraryAccount(String firstName, String`

`lastName, String username, String password) throws SQLException` (see interface description)

For stage 2, this method creates a library account owner object using the parameters values and returns it. For the library account owner id, simply use a hardcoded value.

`public AccountOwner getAccountOwner(int id) throws SQLException` (see interface description)

For stage 2, this method should create one of three possible library account owner objects based on the id provided. The method will compare the id parameter against three id values hardcoded into the method. If the specified id value does not match up with any of the hard coded values, then the method should return **null**. Otherwise, the method should create and return a library account owner object using the given id and hardcoded first and last name values.

`public AccountOwner validateAccountOwner(String username, String password) throws SQLException` (see interface description)

For stage 2, this method should compare the username and password against three username/password values hardcoded into the method. If the specified username/password does not match up with any of the hardcoded values, then the method should return **null**. Otherwise, the method should return one of the three library account owner objects returned by the **AccountOwnerData.getAccountOwner** method (each matching username/password should return a different one of the three library account owners).

The `BadRequestException` Class

Inheritance

This class should inherit from the **ServletException** class. Be sure to override all super class constructors.

The BadRequestException Page Handler

Create an error entry in the **web.xml** file to redirect the exception **BadRequestException** to the error page **badrequest.jspx**.

The `RedirectFilter` Class

Use the ca.on.senecac.prg556.common.RedirectFilter to redirect (client-side redirection) all context root requests to **showmaterialreservations.jspx**.

The `AccountAccessFilter` Class

Create an AccountAccessFilter that checks all jspx page requests to ensure that the library account owner is logged in and if not, redirects to the **librarylogin.jspx** page. For stage 2, the filter simply continues along the filter chain. But be sure to include url pattern for all jspx pages.

The `LimaraRequestListener` Class

This class implements the **ServletRequestListener** interface. When a request is being initialized, the request listener deletes all expired library material reservations. Don't forget to add a **<listener>** entry into the **web.xml** file.

The `LibraryLoginControl` Class

Page Association

The **LibraryLoginControl** class should be associated with the **librarylogin.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

The method should get **username** and **password** from HTML form parameters. If the request method is POST and the parameters are not **null** or empty string, the method should validate the library account owner and, if validation is successful, redirect to the context root. Otherwise, the method should continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `NewLibraryAccountControl` Class

Page Association

The **NewLibraryAccountControl** class should be associated with the **newlibraryaccount.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

If the request method is POST and the non library account owner submitted using the create new library account submit button, the method should make the following checks:

- Get the first name from an HTML form parameter and make sure the first name is not null or empty. If the first name is valid, then it should be saved to a request-scoped attribute.
- Get the last name from an HTML form parameter and make sure the last name is not null or empty. If the last name is valid, then it should be saved to a request-scoped attribute.
- Get the username from an HTML form parameter and make sure the username is not null or empty. If the username is valid, then it should be saved to a request-scoped attribute.
- Get the password from an HTML form parameter and make sure the password is not null or empty. The password should not be saved to a request-scoped attribute.

For each parameter value retrieved and sent back to the page in request-scoped attributes, be sure to **StringHelper.XmlEscape** the text first. If all of the above checks are successful, then the method should check that the password and the confirm password are the same. If there were no validation errors, the method should create a new library account and then redirect to the context root. Otherwise the method should continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `ShowMaterialReservationsControl` Class

Page Association

The **ShowMaterialReservationsControl** class should be associated with the **showmaterialreservations.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

> This method should get a list of all library material reservations for the library account owner id corresponding to one of the library account owners hardcoded in the **AccountOwnerData.getAccountOwner** method and then save the list into a request-scoped attribute (to be used by the **showmaterialreservations.jspx** page). The method should then continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

newlibraryaccount.jspx Page Changes

The **newlibraryaccount.jspx** page must now retrieve the first name, last name, and username from request-scoped attributes and incorporate them as default values for the form text inputs. To set a default value for a form text input, simply set the **value** attribute of the **input** element to be the desired default value.

showmaterialreservations.jspx Page Changes

Since the MaterialReservationData class has been stubbed to always provide two library material reservations, the **showmaterialreservations.jspx** page must be modified to show the title, type, and expiration date for these two library material reservations. Access the library material reservation information using array notation in the JSP EL (e.g. **`${materialreservations[0].title}`** assuming **materialreservations** is the request-scoped attribute name for the library material reservations list saved by **ShowMaterialReservationsControl**). You do not need to format the expiration date for this stage. By default expriation date will format as a date/time value. The material id in the hidden form input should also come from the library material reservations list request-scoped attribute.

Group B Components

Members of this group are responsible for creating the classes **MaterialReservationDAOFactory** and **MaterialReservationData** and stubbing the methods to provide data. The MVC **Control** classes for this group are also specified. These classes will need to use the **MaterialDAOFactory**, **MaterialReservationDAOFactory**, and **AccountOwnerDAOFactory** classes to create the **MaterialDAO**, **MaterialReservationDAO**, and **AccountOwnerDAO** objects as needed. The classes are described below.

The `MaterialReservationDAOFactory` Class

Create this class in a **data** sub-package under your project package.

Methods

`public static MaterialReservationDAO getMaterialReservationDAO()`
Returns a MaterialReservationDAO object.

The `MaterialReservationData` Class

The `MaterialReservationData` class must implement the methods of the MaterialReservationDAO interface. Create this class in a **data** sub-package under your project package.
This class should not be public (use package level access).

Methods

`public void cancelMaterialReservation(String materialId) throws SQLException` (see interface description)
For stage 2, this method does nothing.

`public void deleteExpiredMaterialReservations() throws SQLException` (see interface description)
For stage 2, this method does nothing.

`public MaterialReservation getMaterialReservation(String materialId) throws SQLException` (see interface description)
For stage 2, if the material id matches one of the material ids returned by the **MaterialReservationData.getMaterialReservations** method, then that library material reservation object is returned. For all other material ids, the method should return **null**.

`public List<MaterialReservation> getMaterialReservations(int accountOwnerId) throws SQLException` (see interface description)
For stage 2, this method should simply create a list of two library material reservations. Simply hardcode the library material reservation data. For the expiration date, take the current date and add 7 days (see the **MaterialReservationData.reserveMaterial** method below) The method should add both library material reservations to a list and return it.

`public boolean isMaterialAvailable(String materialId) throws SQLException` (see interface description)
For stage 2, this method should simply return **true**.

`public MaterialReservation reserveMaterial(int accountOwnerId, String materialId) throws SQLException` (see interface description)
For stage 2, this method simply creates a library material reservation object using the parameters values and returns it. For the expiration date, take the current date and add 7 days. E.g.

```
Date today = new Date();
Timestamp expirationDate = new Timestamp(today.getTime() + EXPIRATION_TIME);
```

where EXPIRATION_TIME is a constant defined as 7*24*60*60*1000 (one week in milliseconds). You can assume that the material id given corresponds to one of the three library materials returned by the **MaterialData.getMaterial** method. Get the title and type from the library material object that corresponds to the given material id and include them in the newly created library material reservation object.

The `LimaraMenuFilter` Class

Create a LimaraMenuFilter that executes for all include requests for **limaramenu.jspx**. For stage 2, the filter should get one of the **AccountOwner** objects hardcoded in the **AccountOwnerData.getAccountOwner** method and save it into a request-scoped attribute to be used by the **limaramenu.jspx** page. The filter should also get the number (i.e., how many) of library material reservations for that library account owner and save that value into a request-scoped attribute. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `LogoutControl` Class

Page Association

The **LogoutControl** class should be associated with the fictitious **logout.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

This method should redirect to the context root path.

The `ReserveMaterialsControl` Class

Page Association

The **ReserveMaterialsControl** class should be associated with the **reservematerials.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

If the request method is POST, the method should make the following checks:

- Get the search library material type from an HTML form parameter and make sure the search library material type is not null or empty. If the search library material type is not null or empty, then it should be saved to a request-scoped attribute.
- Get the search library material title (or partial title) from an HTML form parameter and check if the search library material title is not null or empty. If the search library material title is not null or empty, then it should be saved to a request-scoped attribute.

For each parameter value retrieved and sent back to the page in request-scoped attributes, be sure to **StringHelper.XmlEscape** the text first.

If there were no validation errors, the method should get a list of available library materials based on the search criteria and save the list into a request-scoped attribute. If **All** was specified for the search library material type, use **null** instead. If the search library material title (or partial title) is empty string, use **null** instead. Additionally, if the library account owner submitted using the reserve material submit button, the method should:

- Get the material id from an HTML form parameter. If the material id is **null** or empty string, the method should throw a **BadRequestException**. Otherwise the method should get the **Material** object for the material id. If the material id is not valid (i.e., does not correspond to a library material), the method should throw a **BadRequestException**.
- If the material id is valid, the program should check that the specified library material is available. If it is not available, the method should continue along the filter chain.
- If all of the above checks are successful and the library material is available, the method should reserve the library material using the material id specified. For this stage, hardcode the library account owner id to one of the ids used in the **AccountOwnerData.getAccountOwner** method. The method should then redirect to the context root.
- Since a library material can only be reserved by one library account owner, in the time between the check that the library material is available and the reservation of the library material, no other library account owner should be able to reserve the same library material. This can be accomplished using a synchronized block.

If there were validation errors, the method should continue along the filter chain. If the request method is not POST, the method should continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `CancelReservationConfirmationControl` Class

Page Association

The **CancelReservationConfirmationControl** class should be associated with the **cancelreservationconfirmation.jspx** page.

Methods

`String doLogic(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`

The method should get the material id from an HTML form parameter. If the material id is **null** or not a valid number, the method should throw a **BadRequestException**. Otherwise the method should get the **MaterialReservation** object for the material id. If the material id is not valid (i.e., does not correspond to a valid library material reservation object), the method should throw a **BadRequestException**.

a. If the request method is POST
    - If the library account owner submitted using the cancel library material reservation submit button from the **showmaterialreservations.jspx** page, the method should save the library material reservation into a request-scoped attribute (to be used by the **cancelreservationconfirmation.jspx** page). The method should then continue along the filter chain.
    - If the library account owner submitted using the confirm library material reservation cancellation submit button from the **cancelreservationconfirmation.jspx** page, the method should cancel the library material reservation with the material id determined

earlier and then redirect to the context root.
    - If the customer submitted using the abort library material reservation cancellation submit button from the **cancelreservationconfirmation.jspx** page, the method should redirect to the context root.

b. If the method is not POST, the method should redirect to the context root.

If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

limaramenu.jspx Page Fragment Changes

The **limaramenu.jspx** page fragment must now retrieve the number of library material reservations from a request-scoped attribute (saved by **LimaraMenuFilter**) and the library account owner's full name (first name and last name) from a request-scoped attribute which contains an **AccountOwner** object (also saved by **LimaraMenuFilter**) and incorporate them into the menu where required.

reservematerials.jspx Page Changes

The **reservematerials.jspx** page must now retrieve the search title (or partial title) from a request-scoped attribute and incorporate it as a default value for the form text input. To set a default value for a form text input, simply set the **value** attribute of the **input** element to be the desired default value.

Since the MaterialData class has been stubbed to always provide two available library materials, the **reservematerials.jspx** page must be modified to show the title and type for these two library materials. Access the library material information using array notation in the JSP EL (e.g. **`${availableMaterials[0].title}`** assuming **availableMaterials** is the request-scoped attribute name for the available library materials list saved by **ReserveMaterialsControl**). The search library material title (or partial title) and search library material type in the hidden form inputs should also come from the search library material title and search library material type request-scoped attributes, and the material id in the hidden form input should come from the available library materials list request-scoped attribute.

cancelreservationconfirmation.jspx Page Changes

The **cancelreservationconfirmation.jspx** page must now retrieve the material id, the library material title, and the library material type from a **MaterialReservation** object provided through a request-scoped attribute by **CancelReservationConfirmationControl**) and incorporate them into the cancel library material reservation confirmation page wherever they are required.

Back to Overview Stage 1 Stage 3